Machine Learning APIs

Google Cloud Vision, Speech, Translate
Natural Language, Video Intelligence
Common applications

- Optical character recognition
- Language translation
- Automatic image tagging
- Autonomous vehicles
- Video recommendations
- Speech to text
An enormous space

- Many different models and approaches
  - Expert systems / decision trees / knowledge engineering (medical diagnosis)
  - Supervised learning (Bayesian filters for spam detection)
  - Unsupervised learning (Clustering algorithms for Google News)
  - Combinatorial search (Chess)
  - Reinforcement learning (NPC in games)
  - Evolutionary/genetic algorithms (Smart fuzzing)
- Often combined with each other
- But recently…neural networks with supervised learning
  - Convolutional neural networks (CNNs)
Neural networks approach

- Model the way the brain works via large collections of simulated neurons

- Formation and pruning of selected connections encodes information
Simulated in computers

- Selected weights within a simulated network of neurons encodes information
Everything old…

- Neural Networks circa 1970s
  - Ran on original Macintosh!
- Why the renaissance?
  - Video cards with massive numbers of processing units (thanks to gaming)
  - Massive storage capacities for data
  - Crowd-sourced platforms to provide labeling (to learn by example)
ImageNet (2009)

- Pioneered "Deep Learning"
  - Use convolutional neural networks with massive sets of labeled data to solve the computer vision problem
  - Will cover mechanism during finals week
- Approach
  - Ignore academic skeptics and take risk on ancient algorithm
  - Collect image data from the Internet
  - Hire humans to label it via Mechanical Turk
  - Send through enormous neural network
- Profit?
  - Fei Fei Li [https://youtu.be/40riCqvRoMs?t=2m46s](https://youtu.be/40riCqvRoMs?t=2m46s)
Now…“The IMAGENET of x”

- **SpaceNet**
  DigitalGlobe, CosmiQ Works, NVIDIA

- **MusicNet**
  J. Thickstun et al, 2017

- **Medical ImageNet**
  Stanford Radiology, 2017

- **ShapeNet**
  A.Chang et al, 2015

- **EventNet**
  G. Ye et al, 2015

- **ActivityNet**
  F. Heilbron et al, 2015

- But, not all sunshine and rainbows…
Corpus can have issues

- US vs. Russian tanks in 1980s with early NN
  - US images crisp marketing shots on sunny days
  - Russian images grainy and on cloudy days
  - ML trains on wrong feature (crisp/sunny vs. grainy/cloudy)
Corpus can have issues

- Labeling "vacation" photos, a relative task

Garbage In, Garbage Out: machine learning has not repealed the iron law of computer science

- SE Asian workers labeling conference reception photos in hotels as the ideal vacation!
  - Perhaps the beach is hard work if they fish for a living?
- Using ImageNet photos taken by humans on high-quality cameras for drone applications with poor hardware and no human
Bias in labeling

- Institutionalize bias behind facade of an "objective" algorithm
- Racial, socio-economic bias in data being used
  - [GCP podcast #114](#) (2/2018)
  - [FAT* conference](#)
- Example
  - ML for predictive policing trained on domestic violence reports
  - Most common way is via neighbor complaint
  - ML incorrectly learns that only those who live in row-homes and apartments commit domestic violence!
- Hard problem
  - "It’s not always so obvious ahead of time what the bad outcomes might be"
Rare problematic cases

- Self-driving cars
  - Kangaroos, white trucks against a white sky
  - Need millions of miles trained to get enough anomalous conditions
  - Must also train in winter/spring, in snow/rain/clear conditions

---

**Self-Driving Car Technology Foiled By Kangaroos**

Volvo says hopping throws off the vehicles' animal detection system.

---

**Tesla death smash probe: Neither driver nor autopilot saw the truck**

---

Portland State University CS 410/510 Internet, Web, and Cloud Systems
Rare problematic cases

- Successful spear-phishing attacks rare!
  - 372 million e-mails, but only 10 known cases of attacks
    - 99.99% accurate ML algorithm leads to 372k false positives
- Must apply the appropriate learning algorithm
  - Unsupervised learning with anomaly detection
Resistance to adversaries

- Both in training and in inference
- Generative Adversarial Networks

“A panda” 57.7% confidence

“A gibbon” 99.3% confidence
Requires participation for performance

- In thousands of entries

<table>
<thead>
<tr>
<th>Year</th>
<th>Number of Entries</th>
<th>Classification Errors (top-5)</th>
<th>Average Precision For Object Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>35</td>
<td>0.28</td>
<td>1/2</td>
</tr>
<tr>
<td>2011</td>
<td>20</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>29</td>
<td>0.23</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>157</td>
<td>0.66</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td></td>
<td>0.03</td>
</tr>
<tr>
<td>2016</td>
<td></td>
<td></td>
<td>0.03</td>
</tr>
</tbody>
</table>
“Datasets—not algorithms—might be the key limiting factor to development of human-level artificial intelligence.”

ALEXANDER WISSNER-GROSS

Edge.org, 2016
Towards crowd-sourced ML

We’re passing the baton to **Kaggle**: a community of more than 1M data scientists.

Why? **democratizing data** is vital to **democratizing AI**.

image-net.org remains live at Stanford.
An Explosion of Datasets

kaggle

1627 Hosted Datasets
276 Commercial Competitions
1919 Student Competitions
1MM Data Scientists
4MM ML Models Submitted
Question

- Who on this planet has the largest, most interesting, data-sets?
Software's new application building blocks

- Don't build your own when you can use pre-trained models done by experts on massive datasets
  - Part of an emerging API development platform
  - All with REST APIs for access and results via JSON
  - Abstraction raised, “Hello world” will never be the same
Cloud Vision API

- Image recognition
  - Image labeling on thousands of labels
  - Face detection
  - Sentiment analysis (Emotobooth)
  - Text detection (optical character recognition)
  - SafeSearch content identification (adult/violent content)
  - Logo identification
  - Landmark identification
Cloud Speech APIs

- Speech-to-Text
  - Word recognition
  - Context-aware transcription
  - Automated punctuation
  - Offensive content detection

- Also, Text-to-Speech (speech synthesis)

- Both in 120 languages
Cloud Translation API

- Language detection and translation
  - 100+ languages
  - Python, Java, Ruby, Objective-C bindings via Google API client libraries

Cloud Natural Language API

- Language analysis
  - Syntax analysis
  - Semantic analysis, entity recognition
  - Sentiment analysis
  - Common e-mail responses
Cloud Video Intelligence API

- Summary and information extraction from video
  - Autotag objects in video to enable searching
  - Scene detection for thumbnail generation
  - Automated highlights (RedZone!)
  - Trained on massive set of labeled YouTube videos
- Demo
  - [https://youtu.be/Rggggd018?t=14m30s](https://youtu.be/Rggggd018?t=14m30s) (14:30-17:40)
Cloud Video Intelligence API

- Demo code
  - Flow chart triggered when new video placed in bucket
  - https://github.com/sararob/video-intelligence-demo
Cloud AutoML (2018)

- ML services (e.g. Cloud Vision API) trained on general datasets
  - Custom images and domain labels often needed
- AutoML
  - Apply transfer learning to re-use trained models to quickly learn new domains
  - Custom models via
    - Labeled data (done by user)
    - Unlabeled data (done by humans at Google)
- Example: Custom models to recognize machine parts
- Democratizes ML for the masses
  - All that is required is for you to upload your data with labels or exemplars for Google to label
Putting APIs together

- Univision live video broadcasts
- Video capture, transcoding, distribution
  - "YouTube infrastructure" as a service
  - YouTube code for video capture, encoding, and distribution
- Real-time close-captioning
  - Speech API for transcription
  - Translate API to support multiple languages
- Targeted in-stream advertising
  - ML Engine, ML APIs
    - Video Intelligence, Vision, and Natural Language Processing APIs
    - Determining the context of video
  - AdSense
    - # of ads, length of each ad, content of ad
    - Done per user via user analytics and context
- Automatic clipping and tagging of video for thumbnails, trailers, and highlights
  - Video Intelligence and Vision APIs
Example: unCAPTCHA

- Audio version for visually impaired
- Broken with Machine Learning
  - USENIX WOOT ’17

unCaptcha: A Low-Resource Defeat of reCaptcha's Audio Challenge

Authors:
Kevin Bock, Daven Patel, George Hughey, and Dave Levin, University of Maryland

- Use free tier of Google Cloud Speech API, IBM BlueMix, AWS to solve
- Irony?
  - Cloud Speech API protected by reCAPTCHA
  - System is self-sufficient!
Labs
ML APIs Lab #1

- Integrating Machine Learning APIs (25 min)
Enable APIs

- Skip project creation step (use your course project)
- Ensure Cloud Speech, Cloud Translation, and Cloud Natural Language Processing APIs are enabled
  - As done with Cloud Vision previously (via web console)
  - Or via individual `gcloud` commands within Cloud Shell

```
gcloud services enable speech.googleapis.com
```
```
gcloud services enable translate.googleapis.com
```
```
gcloud services enable language.googleapis.com
```
```
gcloud services enable vision.googleapis.com
```
Setup

- Create a service account for you to access ML APIs
  - Note I've chosen to name it cs410mlapis
  - Bound to your PROJECT_ID and placed in a JSON file

```bash
cd $HOME

gcloud iam service-accounts create cs410mlapis

gcloud iam service-accounts keys create cs410mlapis.json --iam-account cs410mlapis@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com

export GOOGLE_APPLICATION_CREDENTIALS=$HOME/cs410mlapis.json
```

- Should get message that [cs410mlapis@cs410c-odinid.iam.gserviceaccount.com](mailto:cs410mlapis@cs410c-odinid.iam.gserviceaccount.com) created
- Set environment variable to point to your created credential file

```bash
export GOOGLE_APPLICATION_CREDENTIALS=$HOME/cs410mlapis.json
```

- Note that this will only set the variable for your current session.
- Update `~/.bashrc` to set credentials for each Cloud Shell session

```bash
export GOOGLE_APPLICATION_CREDENTIALS=$HOME/cs410mlapis.json
source /google/devshell/bashrc.google
```
Cloud Vision via Python

- If you haven’t downloaded the Python samples, do so

```bash
$ git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git
```

- Install Cloud Vision package

```bash
$ pip install --upgrade google-cloud-vision --user
```

- Go to Vision cloud-client code

```bash
$ cd ~/python-docs-samples/vision/cloud-client/detect
```

- Run a detection that returns the labels generated with an image of a bird given its URI, show the output

```bash
$ python detect.py labels-uri gs://ml-api-codelab/birds.jpg
```
Cloud Vision via Python

• Examine `detect.py`

```python
from google.cloud import vision

def detect_labels_uri(uri):
    """Detects labels in the file located in Google Cloud Storage or on the Web."""
    client = vision.ImageAnnotatorClient()
    image = vision.types.Image()
    image.source.image_uri = uri

    response = client.label_detection(image=image)
    labels = response.label_annotations
    print('Labels:"

    for label in labels:
        print(label.description)
```

• Then, use the script to run a detection on a logo you find on the Internet via its URI
Cloud Speech via Python

- Install Cloud Speech package
  
  ```bash
  pip install --upgrade google-cloud-speech --user
  ```

- Go to Speech cloud-client code
  
  ```bash
  cd ~/python-docs-samples/speech/cloud-client
  ```

- Fix line 73 of `transcribe.py` (delete encoding and `sample_hertz_rate`, set language to `tr-TR`)

```python
def transcribe_gcs(gcs_uri):
    """Transcribes the audio file specified by the gcs_uri."""
    from google.cloud import speech
    from google.cloud.speech import enums
    from google.cloud.speech import types
    client = speech.SpeechClient()

    audio = types.RecognitionAudio(uri=gcs_uri)
    config = types.RecognitionConfig(language_code='tr-TR')

    response = client.recognize(config, audio)
    # Each result is for a consecutive portion of the audio. Iterate through
    # them to get the transcripts for the entire audio file.
    for result in response.results:
        # The first alternative is the most likely one for this portion.
        print(u'Transcript: {}', format(result.alternatives[0].transcript))
```
Cloud Speech via Python

- **Run** `transcribe.py` **on the given URI and show output**

```
python transcribe.py gs://ml-api-codelab/tr-ostrich.wav
```
Cloud Translate via Python

- Install Cloud Translate package
  ```bash
  pip install --upgrade google-cloud-translate --user
  ```
- Go to Speech cloud-client code
  ```bash
  cd ~/python-docs-samples/translate/cloud-client
  ```
- Examine code

```python
from google.cloud import translate
import six

def translate_text(target, text):
    """Translates text into the target language.
    See https://g.co/cloud/translate/v2/translate-reference#supported_languages
    """
    translate_client = translate.Client()

    if isinstance(text, six.binary_type):
        text = text.decode('utf-8')

    result = translate_client.translate(
        text, target_language=target)

    print(u'Text: {}').format(result['input']))
    print(u'Translation: {}').format(result['translatedText']))
    print(u'Detected source language: {}').format(
        result['detectedSourceLanguage']))
```
Cloud Translate via Python

- Run `snippets.py` on the text string and show output

```
python snippets.py translate-text en '你有沒有帶外套'
```
Cloud Natural Language via Python

- Install Cloud Natural Language package
  
  ```sh
  pip install --upgrade google-cloud-language --user
  ```

- Go to Speech cloud-client code
  
  ```sh
  cd ~/python-docs-samples/language/cloud-client/v1
  ```
Cloud Natural Language via Python

- Examine code for sentiment analysis

```python
from google.cloud import language
from google.cloud.language import enums
from google.cloud.language import types
import six

def sentiment_text(text):
    """Detects sentiment in the text."""
    client = language.LanguageServiceClient()

    if isinstance(text, six.binary_type):
        text = text.decode('utf-8')

    # Instantiates a plain text document.
    document = types.Document(
        content=text,
        type=enums.Document.Type.PLAIN_TEXT)

    # Detects sentiment in the document. You can also analyze HTML with:
    # document.type == enums.Document.Type.HTML
    sentiment = client.analyze_sentiment(document).document_sentiment

    print('Score: {}
print('Magnitude: {}'.format(sentiment.score), format(sentiment.magnitude))
```
Cloud Natural Language via Python

- Examine code for entity analysis

```python
def entities_text(text):
    """Detects entities in the text.""
    client = language.LanguageServiceClient()

    if isinstance(text, six.binary_type):
        text = text.decode('utf-8')

    document = types.Document(
        content=text,
        type=enums.Document.Type.PLAIN_TEXT)

    entities = client.analyze_entities(document).entities

    # entity types from enums.Entity.Type
    entity_type = ('UNKNOWN', 'PERSON', 'LOCATION', 'ORGANIZATION',
                   'EVENT', 'WORK_OF_ART', 'CONSUMER_GOOD', 'OTHER')

    for entity in entities:
        print('=' * 20)
        print(u'{{:<16}}: {}'.format('name', entity.name))
        print(u'{{:<16}}: {}'.format('type', entity_type[entity.type]))
        print(u'{{:<16}}: {}'.format('metadata', entity.metadata))
        print(u'{{:<16}}: {}'.format('salience', entity.salience))
        print(u'{{:<16}}: {}'.format('wikipedia_url',
                              entity.metadata.get('wikipedia_url', '-')))```
Cloud Natural Language via Python

- Run `snippets.py` on the text string and show output
  ```
  python snippets.py entities-text 'where did you leave my bike'
  ```

- For fun, also try sentiment analysis
  ```
  python snippets.py sentiment-text 'homework is awful!'
  python snippets.py sentiment-text 'homework is awesome?'
  python snippets.py sentiment-text 'homework is awesome.'
  python snippets.py sentiment-text 'homework is awesome!' 
  ```
Integration

- See if words in a recording describe an object in an image
- Previous calls modified to return results as text (vs. print)
  - Audio transcription to translation to NLP to obtain entities
  - Image analysis to obtain labels
- Comparison to determine match
  - https://github.com/googlecodelabs/integrating-ml-apis

![Diagram of the process]

In foreign language

Compare
- tr-TR speech samples:
  - gs://ml-api-codelab/tr-ball.wav
  - gs://ml-api-codelab/tr-bike.wav
  - gs://ml-api-codelab/tr-jacket.wav
  - gs://ml-api-codelab/tr-ostrich.wav

- de-DE speech samples:
  - gs://ml-api-codelab/de-ball.wav
  - gs://ml-api-codelab/de-bike.wav
  - gs://ml-api-codelab/de-jacket.wav
  - gs://ml-api-codelab/de-ostrich.wav
Integration

- See code for mods to `transcribe_gcs()` (Speech), `translate_text()` (Translate), `entities_text()` (Natural Language) and `detect_labels_uri()` (Vision)

```python
def compare_audio_to_image(language, audio, image):
    """Checks whether a speech audio is relevant to an image."""
    # speech audio -> text
    transcription = transcribe_gcs(language, audio)

    # text of any language -> english text
    translation = translate_text('en', transcription)

    # text -> entities
    entities = entities_text(translation)

    # image -> labels
    labels = detect_labels_uri(image)

    # naive check for whether entities intersect with labels
    has_match = False
    for entity in entities:
        if entity in labels:
            print(f'The audio and image both contain: {entity}.')
            has_match = True
        if not has_match:
            print('The audio and image do not appear to be related.
```
ML APIs Lab #1

- Run at least 3 pairs other than the one given in the walkthrough

```
python solution.py tr-TR gs://ml-api-codelab/tr-ball.wav gs://ml-api-codelab/football.jpg
```

- Integrating Machine Learning APIs (25 min)
  - [https://codelabs.developers.google.com/codelabs/cloud-ml-apis](https://codelabs.developers.google.com/codelabs/cloud-ml-apis)
ML APIs Lab #2

- Using the (rest of the) Vision API with Python (8 min)
  - Optical Character Recognition (OCR) (text detection)
  - Landmark detection
  - Sentiment analysis (face detection)
ML APIs Lab #2

- Skip Steps 2, 3, and 4. (Re-use setup from ML API Lab #1)
- Copy image files into your own bucket, then set all files in bucket to publicly readable

```bash
gsutil cp gs://cloud-vision-codelab/otter_crossing.jpg gs://$DEVSHELL_PROJECT_ID
gsutil cp gs://cloud-vision-codelab/eiffel_tower.jpg gs://$DEVSHELL_PROJECT_ID
gsutil cp gs://cloud-vision-codelab/face_surprise.jpg gs://$DEVSHELL_PROJECT_ID
gsutil cp gs://cloud-vision-codelab/face_no_surprise.png gs://$DEVSHELL_PROJECT_ID
gsutil -m acl set -R -a public-read gs://$DEVSHELL_PROJECT_ID
```

- For the rest of the examples, my project ID is used in the `gs://` URLs, use yours instead

  ```bash
  $ echo $DEVSHELL_PROJECT_ID
  cs410c-wuchang-201515
  ```
ML APIs Lab #2

• Launch interactive *ipython* to perform lab

```
wuchang@cloudshell:~ (cs410c-wuchang-201515)$ ipython ...
...
...
In [1]:
```
• Show full image of the Otter Crossing sign via your bucket
• Then use Vision's `text_detection()` to perform an OCR operation on a picture of the sign (substitute your bucket name in the `gs://` URL)

```python
from google.cloud import vision
from google.cloud.vision import types
client = vision.ImageAnnotatorClient()
image = vision.types.Image()
image.source.image_uri = 'gs://cs410c-wuchang-201515/otter_crossing.jpg'
resp = client.text_detection(image=image)
print('
'.join([d.description for d in resp.text_annotations]))
```
• Show full Eiffel Tower image in your bucket
• Then use Vision's `landmark_detection()` to identify it of famous places (substitute your bucket name in the `gs://` URL)

```python
from google.cloud import vision
from google.cloud.vision import types

client = vision.ImageAnnotatorClient()
image = vision.types.Image()
image.source.image_uri = 'gs://cs410c-wuchang-201515/eiffel_tower.jpg'
resp = client.landmark_detection(image=image)
print(resp.landmark_annotations)
```
• Show the two face images in your bucket
• Then use Vision's `face_detection()` to annotate images (substitute your bucket name in the `gs://` URL)
• See the likelihood of the faces showing surprise

```python
from google.cloud import vision
from google.cloud.vision import types

client = vision.ImageAnnotatorClient()
image = vision.types.Image()
likelihood_name = ('UNKNOWN', 'VERY_UNLIKELY', 'UNLIKELY', 'POSSIBLE', 'LIKELY', 'VERYLIKELY')
for pic in ('face_surprise.jpg', 'face_no_surprise.png'):
    image.source.image_uri = 'gs://cs410c-wuchang-201515/' + pic
    resp = client.face_detection(image=image)
    faces = resp.face_annotations
    for face in faces:
        print(pic + ': surprise: {}\n'.format(likelihood_name[face.surprise_likelihood]))
```
ML APIs Lab #2

- Using the Vision API with Python (8 min)
  - [https://codelabs.developers.google.com/codelabs/cloud-vision-api-python](https://codelabs.developers.google.com/codelabs/cloud-vision-api-python)
ML APIs Lab #3

- Deploying a Python Flask Web Application to App Engine Flexible (24 min)
- Skip steps 2, 3, and 5 (already done previously)
- Do Step 4 In Cloud Shell

```
cd python-docs-samples/codelabs/flex_and_vision
```
For Step 6, we will re-use the service account from ML APIs Lab #1

- Skip all steps except the one that gives project owner access to service account created in ML APIs Lab #1
- Use command below since we named our account cs410mlapis (not codelab)

```bash
gcloud projects add-iam-policy-binding ${DEVSHHELL_PROJECT_ID} --member serviceAccount:cs410mlapis@${DEVSHHELL_PROJECT_ID}.iam.gserviceaccount.com --role roles/owner
```
• Create python3 environment
  
  ```
  virtualenv -p python3 env
  source env/bin/activate
  pip install -r requirements.txt
  ```

• Set the location of the Cloud Storage bucket for the app's images via an environment variable
  • If you have deleted your gs://${DEVSHELL_PROJECT_ID} bucket, create it again, then set the environment variable
    ```
    export CLOUD_STORAGE_BUCKET=${PROJECT_ID}
    ```

• Deploy app
  ```
  gcloud app deploy
  ```
• Test the app via the web preview

  - Preview on port 8080
  - Change port
  - About web preview

• Upload a photo to detect joy in faces

  - Joy Likelihood for Face: Very Likely
  
  Untitled.jpg was uploaded 2018-05-28 09:53:40.511135-00:00.

  This Python Flask application demonstrates App Engine Flexible, Google Cloud Storage, Datastore, and the Cloud Vision API.
Code for default route

```python
@app.route('/

def homepage():
    # Create a Cloud Datastore client.
    datastore_client = datastore.Client()

    # Use the Cloud Datastore client to fetch
    # information from Datastore about each photo
    query = datastore_client.query(kind='Faces')
    image_entities = list(query.fetch())

    # Pass image_entities to Jinja2 template to render
    return render_template('homepage.html',
                           image_entities=image_entities)
```
Code for uploading new images

- `upload_photo()`

```python
from google.cloud import datastore
from google.cloud import storage
from google.cloud import vision

CLOUD_STORAGE_BUCKET = os.getenv('CLOUD_STORAGE_BUCKET')

@app.route('/upload_photo', methods=['GET', 'POST'])
def upload_photo():
    photo = request.files['file']  # File from form submission

    storage_client = storage.Client()  # Create storage client.

    # Get bucket
    bucket = storage_client.get_bucket(CLOUD_STORAGE_BUCKET)

    # Create blob to store uploaded content then upload content to it
    blob = bucket.blob(photo.filename)
    blob.upload_from_string(photo.read(),
                            content_type=photo.content_type)

    # Make blob publicly available
    blob.make_public()
```
# Create a Cloud Vision client.
vision_client = vision.ImageAnnotatorClient()

# Use the Cloud Vision client to detect a face for our image.
source_uri = 'gs://{}{}'.format(CLOUD_STORAGE_BUCKET, blob.name)
image = vision.types.Image(
    source=vision.types.ImageSource(gcs_image_uri=source_uri))

faces = vision_client.face_detection(image).face_annotations

# If face detected, store likelihood that the face displays 'joy' based on Vision's annotations
if len(faces) > 0:
    face = faces[0]  # Convert the likelihood string.
    likelihoods = ['Unknown', 'Very Unlikely', 'Unlikely', 'Possible', 'Likely', 'Very Likely']
    face_joy = likelihoods[face.joy_likelihood]
else:
    face_joy = 'Unknown'

# Code for getting face annotations
Code for insertion to datastore

- **upload_photo()**

```python
datastore_client = datastore.Client()  # Create datastore client.
current_datetime = datetime.now()     # Fetch current date / time.

kind = 'Faces'                        # Set kind for new entity.
name = blob.name                      # Set name/ID for new entity.

# Create the Cloud Datastore key for the new entity.
key = datastore_client.key(kind, name)

# Construct the new entity using the key as a dictionary
# including "face_joy" label from Cloud Vision face detection
entity = datastore.Entity(key)
entity['blob_name'] = blob.name
entity['image_public_url'] = blob.public_url
entity['timestamp'] = current_datetime
entity['joy'] = face_joy

# Save the new entity to Datastore.
datastore_client.put(entity)

return redirect('/')                 # Return redirect to root directory.
```
App Engine configuration

- Change `app.yaml` to point to your storage bucket

```yaml
runtime: python
env: flex
entrypoint: gunicorn -b :$PORT main:app

runtime_config:
  python_version: 3
env_variables:
  CLOUD_STORAGE_BUCKET: cs410c-wuchang-201515
```
Deploy app

- Deactivate development environment
  ```
degactivate
  ```
- Deploy
  ```
gcloud app deploy
  ```
- Note the custom container built and pushed into gcr.io to support the flexible environment's deployment onto App Engine
- Show application running at
  ```
  https://<PROJECT_ID>.appspot.com
  ```
ML APIs Lab #3

- Deploying a Python Flask Web Application to App Engine Flexible (24 min)
  - [https://codelabs.developers.google.com/codelabs/cloud-vision-app-engine](https://codelabs.developers.google.com/codelabs/cloud-vision-app-engine)
Final project

• Adapt your web application from homework assignments to utilize a Machine Learning API of your choice (Vision, Natural Language Processing, Speech, Video Intelligence, etc.)
  • Code submitted in directory called `final`
  • Include source code and a file `url.txt` with a link to a running version of your application.
  • Upload a narrated screencast to MediaSpace channel [https://media.pdx.edu/channel/2018Spring_InternetCloudSystems/89429282](https://media.pdx.edu/channel/2018Spring_InternetCloudSystems/89429282) that includes
    • A demo of the application on Google Cloud
    • A code walk-through that explains how the application works
    • A detailed discussion of which code was contributed by which person if work done as a group.
• Rubric
  • Overall functionality
  • Code documentation (such as Docstrings, comments)
  • Code readability and modularity
  • Thoroughness of walkthrough (including setup)
  • git repository activity (commits, commit messages, tags)
  • Discussion of contributions of each member (if applicable)
Extra
ML APIs Lab #3

- Enable APIs if necessary (already done most likely)

```
gcloud services enable vision.googleapis.com

gcloud services enable storage-component.googleapis.com

gcloud services enable datastore.googleapis.com
```
ML APIs Lab alternatives

- **Cloud Speech API**
  - Speaking with a Webpage (55 min)
    - [https://codelabs.developers.google.com/codelabs/speaking-with-a-webpage/](https://codelabs.developers.google.com/codelabs/speaking-with-a-webpage/) (Java, Javascript)
  - Speech to Text Transcription (14 min)
    - [https://codelabs.developers.google.com/codelabs/cloud-speech-intro](https://codelabs.developers.google.com/codelabs/cloud-speech-intro) (JSON, curl)
  - Google Cloud Speech API : Node.js example (21 min)
    - [https://codelabs.developers.google.com/codelabs/cloud-speech-nodejs/](https://codelabs.developers.google.com/codelabs/cloud-speech-nodejs/) (Javascript)

- **Cloud Translation API**
  - Translate Text with the Translation API (27 min)

- **Cloud Vision API**
  - Using Cloud Vision with Node.js (48 min)
    - [https://codelabs.developers.google.com/codelabs/cloud-vision-nodejs](https://codelabs.developers.google.com/codelabs/cloud-vision-nodejs) (Javascript)
  - Detect Labels, Faces, and Landmarks in Images with the Cloud Vision API (21 min)
    - Can also do logos, safe search, and text
    - [https://codelabs.developers.google.com/codelabs/cloud-vision-intro/](https://codelabs.developers.google.com/codelabs/cloud-vision-intro/) (JSON, curl)

- **Cloud Natural Language API**
  - Entity, Sentiment, and Syntax Analysis with the Natural Language API (23 min)
    - [https://codelabs.developers.google.com/codelabs/cloud-nl-intro/](https://codelabs.developers.google.com/codelabs/cloud-nl-intro/) (JSON, curl)
  - Classify Text into Categories with the Natural Language API (23 min)
    - [https://codelabs.developers.google.com/codelabs/cloud-nl-text-classification](https://codelabs.developers.google.com/codelabs/cloud-nl-text-classification) (JSON, curl, python)

- **Multi**
  - Try Vision, Translate, Natural Language API, Speech API (15 min + github walkthrough)
    - [https://codelabs.developers.google.com/codelabs/cpb100-translate-api/](https://codelabs.developers.google.com/codelabs/cpb100-translate-api/) (iPython notebook)